1. Introduction

Until recently, most government web sites were constructed much like any commercial web site. They used a table-based layout with tables nested within the layout table, javascript effects, tags that had been around since HTML 2.0, and more attention to the "gee-whiz" factor than anything else.

Since there were many differences between browsers, some developers created a couple of different versions of the web site and switched users to a certain version depending on what type of browser was used. All this made for very large pages and multiple opportunities for things to go wrong.

As web browsers have matured to comply with World Wide Web Consortium (W3C) standards, however, the time is right for web developers to begin writing to those standards. This will ultimately make web development and maintenance easier and help Missouri government web developers to reach all their potential users.

Missouri government web sites have certain standards that they must follow. The main standard is accessibility—making sure that all constituents can read the content of the site, regardless of physical abilities or browser preference. However, another goal has to be usability—how easy will it be for a constituent to find the information she is looking for? A third objective is integration with the state portal—can the user tell that the site is an official Missouri government site? Does the page link back to the state home page?

The DMD believes that following the W3C Web Standards, Missouri government web developers will meet most of the accessibility standards without having to resort to expensive third-party tools to create accessible versions of their web sites.

These standards and guidelines have been used by many Missouri government agencies and serve as a "best practices" guide for other state agencies to develop effective web sites for the Missouri State Web.

Using these guidelines will help agencies:

- Develop accessible web pages
- Make more usable sites
- Communicate more effectively
- Make maintenance easier

- To communicate information efficiently and effectively
- To improve the ease of use and organization of information on state Web sites
- To enhance the interoperability of state Web servers and Web sites
- To meet user expectation of information and technology

Scope

The guidelines are concerned with the web developer's final product, i.e., what is rendered on the browser, as opposed to any particular server technology used to produce the final code. Thus, we will not address techniques specific to ASP, .NET, JSP, PHP, CFM, Perl or any other scripting languages.

The guidelines are not limited to just the Internet, though. They apply to all browser-enabled applications, whether on the Internet, an intranet or an extranet. Following the guidelines will help to make sure all these applications meet Missouri standards.

The items presented here tend to fall into one or more of three categories:
1. Common elements
2. Usability
3. Accessibility

Common elements are those items that we recommend that all state web pages have. Examples include navigation tools, links, meta data and icons.

Usability items help users use the site more effectively or help the web developers maintain the site more easily.

Accessibility standards are items that address the OIT accessibility Standards, http://oit.mo.gov/standards/ITGS0003_Missouri_IT_Accessibility_Standards.doc

Authors
This document was drafted by members of the Digital Media Developers (DMD) group, a subcommittee of ITAB composed of state government employees involved in issues relating to Web and digital media development. Principal contributors to this revision were Debbie Boeckman (DNR), Connie Farris (MSHP), Erica Gage (SOS), Debbie Karaff (DSS) and Kevin Lanahan (MDI).

You can find out more about the DMD at http://www.oa.mo.gov/dmd/.

Standards

Missouri government web sites should support the Missouri IT Accessibility Standard, the current Web standard, XHTML 1.0 and Cascading Style Sheets (CSS2). This will involve using techniques in any redesign or redeployment that may be new to some web developers.

If you write your code to standards, your pages will be mostly accessible by default.

Web standards have finally matured. Web Accessibility guidelines have been standardized since 1999, federal law Section 508 has been in effect since 1998, XHTML 1.0 has been a W3C recommended standard since 2000 and XHTML 1.1 (modular XHTML) has been recommended since 2001. CSS1 has been recommended since 1996 and CSS2 since 1998.

This pause in web standards evolution has allowed browsers to implement most of the features of these standards. This means that you, the web developer, can expect a standards-compliant browser to behave in a predictable manner. That means you no longer have to do a lot of work-arounds or browser detection to make sure your sites are viewable in most browsers.

For accessibility, this means that you don't have to use third-party applications to create “text-only” pages for accessibility. By using standards, your web pages are accessible by default.

XHTML (and HTML 4) and CSS allow for separation of content and presentation, which means that any page can be structured very simply, using semantic markup and no presentational markup. All the presentation information is contained in the style sheet, which instructs the browser how to display the tags in the web page.

What this means is that you can create a very simple text-only page and mark it up to display any way you please; users that can't use style sheets will only see the plain text page and users with current browsers will see the pretty page. This is a great step towards accessibility.

Note:
Each state agency has the freedom to design and develop a Web site appropriate to inform and serve their audience. Different agencies have different resources and skills, and they should be free to use them to their best abilities.

However, there are some best practices which can make the development and maintenance of a site easier. In addition, there are common elements that will help brand each web site as being part of the State of Missouri web and make navigation between different agencies easier for the public.

These guidelines place no restrictions on the design and development of a Web site; they are recommendations to ensure a successful deployment of a Web site.

Moving to Standards

Moving to a standards-based format will mean you will have to learn some new things and relearn some old things.

XHTML has a few differences from HTML that are easily learned and a few differences that will have to become habits over time.

Cascading Style Sheets have been around for a few years, but browser support for CSS2 is only a few years old, and you will probably want to get a good book to help explain all CSS can do for you.

Accessibility

Accessibility is not optional; it is mandated by state statute Section 191.863 RSMo. Missouri web sites must follow the Missouri IT Accessibility Standard (http://oit.mo.gov/initiatives/itaccessibility.html), which follows the Federal 508 web accessibility standards with just a couple of exceptions.

The need to make Missouri web sites accessible drives the need to follow standards in web design. A well-designed standards-compliant site will be mostly accessible because it will be a basic, well-formed text page with some presentation styling.

XHTML Rules

There are ten major requirements for XHTML that are not in HTML.

1. You must use a proper DTD.
2. You must use a proper namespace.
3. You must declare your content type.
4. You must make all your tags lowercase.
5. You must quote all attribute values.
6. All attributes require values.
7. You must close all tags.
8. Even empty tags, like <br>.
9. You can't use double dashes in comments, and
10. You need to encode all < and & characters.

Document Type Definition (DTD)

Each XHTML page should have a Document Type Definition (DTD or Doctype) as the first line of the page.

The DTD tells the browser what set of tags to interpret in your page. XHTML specifies three DTDs, so authors must include one of the following document type declarations in their documents. The DTDs vary in the elements they support.

〈 The XHTML Strict DTD includes all elements and attributes that have not been deprecated or do not appear in frameset documents. For documents that use this DTD, use this document type declaration:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

〈 The XHTML Transitional DTD includes everything in the strict DTD plus deprecated elements and attributes (most of which concern visual presentation). For documents that use this DTD, use this document type declaration:

```
<!DOCTYPE HTML "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

〈 The XHTML Frameset DTD includes everything in the transitional DTD plus frames as well. For documents that use this DTD, use this document type declaration:

```
<!DOCTYPE HTML "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

If you are using HTML, DTDs can be found at http://www.w3.org/TR/html401/sgml/dtd.html.

The DTDs tell the browser how to interpret the code in your page. XHTML-Transitional is closest to regular HTML, and is most forgiving of invalid code.

Most web editors will either insert a DTD automatically or can be configured so the main template will automatically insert one.

Strict or Transitional?

Transitional (also known as "quirks") mode kicks in whenever a DTD is not declared, or if your document does not validate. It is the default DTD for Internet Explorer 6. Writing to a transitional standard allows you to use some deprecated attributes, like "*bgcolor.*"

Strict mode doesn't cut you any slack. Your code must validate or your page will get displayed using quirks mode. This means you have to be a little more careful when creating your pages, but you will have greater control over your page. IE6 will display strict mode properly, so you will have a consistent look no matter which browser your users use.

So which one do you use? If you are uncomfortable with changing the way you have always created pages, Transitional mode may be the one for you. It will give you a little more time to clean up sloppy code and tighten up your pages. But remember, it is *transitional*. You really need to think about switching over to Strict sooner or later.

If you make the jump into Strict mode, you may be confused early on as you get used to the no-nonsense rules, but they soon become second nature. Some development tools, like Dreamweaver MX, will automatically convert your pages to comply with your DTD.

Namespace

The namespace is an extension of the basic <html> tag found at the top of each old web page. It follows immediately after the DTD statement, and looks like this:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">
```

The first part of this statement, *xmlns="http://www.w3.org/1999/xhtml*", tells the browser where to find the namespace on the web.

The other two parts, *xml:lang="en" lang="en"*, indicate that the XML and page language is English.

Content type

The actual W3C XHTML specification calls for an <xml> statement above the DTD, where you declare the character encoding for the document. Unfortunately, if you do this, IE will automatically render the page in quirks mode, which can mess up your layout. Other browsers may actually crash.

Instead, you will insert a meta tag in the head of the document, like this:

```
<meta http-equiv="content-type" content="text/html;
charset=ISO-8859-1" />
```

In this example, the document uses the ISO-8859-1 character set (also known as Latin-1). There are other character sets available if you are coding in other languages or need an internationalized character set.

Lowercase tags

XHTML is case-sensitive. All elements and attributes (tags and properties) must be in lowercase in order to validate. For example, this is invalid code:

```
<OL CLASS="bigfont"><LI OnMouseOver="doJS">Item
1</LI><LI>Item 2</LI></OL>
```

but this is valid code:

```
<ol class="bigfont"><li onmouseover="doJS">Item
1</li><li>Item 2</li></ol>
```

Quote attribute values

This is pretty simple. Instead of having code like this:

```
<table cellspacing=0 cellpadding=2 border=1>
```

You will need to use quotes:

```
<table cellspacing="0" cellpadding="2" border="1">
```

You also need to separate each attribute with a space.

All attributes require values

Any attribute that was just a word in HTML will need to have a value declared as well.
HTML:

```
      <input type=”radio” name=”radio1” value=”5” selected>
```

XHTML:

```
      <input type=”radio” name=”radio1” value=”5”
      selected=”selected”>
```

Close all tags

HTML let you assume a tag closed when you started a new one, like this:

```
      <p>The quick red fox jumped over the lazy brown dog.
      <p>She sells seashells by the seashore.
```

XHTML makes you close those tags, which is a good habit to get into anyway:

```
      <p>The quick red fox jumped over the lazy brown dog.</p>
      <p>She sells seashells by the seashore.</p>
```

Close the empty tags, too

Tags like `<br>`, `<hr>` and `<img>` do not have a corresponding closing tag (like `</br>`, `</hr>` or `</img>`).

In XHTML, you close the tag by adding a space and a forward slash before the closing >, like this: `<br />`, `<hr />` and `<img />`

No double dashes in comments

XHTML only allows a double dash (“--”) at the beginning and the end of a comment. So this is not allowed:

```
      <!--This is a bad comment—no fooling. -->
```

But you can put a space between the dashes, so this is valid:

```
      <!--This is a valid comment- -for real. -->
```

Encode special characters

If you use an ampersand, greater than sign, less than sign or quote (“&”, “>”, “<” and “””, respectively) in your text, you will need to encode them like this: &amp;, &gt;, &lt; and &quot;.

Those characters are reserved in XML, but XHTML validators will let you off with a warning.

As you create new pages...
Keep the following in mind when creating your pages:

1. Use valid HTML.
2. Content should be separate from style.
3. Don't make pages too long so that there is endless scrolling, unless you provide "anchors" and links to aid the reader in navigation.
4. Avoid using deprecated tags.
   The following tags have been deprecated and should no longer be used. HTML 4.01 will allow you to use them, but XHTML 1.0 forbids them:
   〈 CENTER
   〈 FONT
   〈 U (underline)
   〈 APPLET

These tags should be replaced with appropriate style sheets or standards-compliant tags.

Cascading Style Sheets
A major component of XHTML and HTML 4 standards is the use of Cascading Style Sheets (CSS). CSS allows the Web author to separate web content from web style. By creating a style sheet, you can greatly reduce the amount of in-line styling (FONT tags, tables and link colors, for example), which will, in turn, make updates and redesigns easier.

Advantages of CSS

The first big advantage of CSS is code reduction. All the CSS can be stored in an external style sheet that is stored in cache, so after it initially loads it is available for all subsequent pages. And all the table and presentational code that you currently use would disappear. You can expect a 10-50% reduction in code, depending on how clean your original code is.

Another advantage is ease of update. If you code all your <h1> tags to be 20 pixels high, and later decide to make them 25 pixels, you can change one line of code and the entire site is updated. Much, much easier than tickling through your entire site to change each and every tag by hand.

The third advantage is accessibility. You should be able to turn off style sheets and still read the entire page. That means any user with an obsolete browser or a screen reader should still be able to access all your content.

Disadvantages of CSS

Well, it is confusing at first. The terminology is different and it will take a while to learn what CSS elements and attributes correspond to the obsolete HTML attributes.

It doesn't work in older browsers. IE3, IE4 and Netscape 4x had incomplete CSS compatibility, and earlier browsers (and, of course, non-graphical browsers) do not have any support for CSS at all. So, if you dealing with a large population that only uses obsolete technology or does not use graphical browsers, they may not see your pages in all their styled glory. However, there is a good chance that they can't see your pages at all if you are using font tags and other in-line, non-semantic code.

A third disadvantage is that CSS can be addictive. Once you start working with it, you just can't stop.

Validation
Three words to ensure the success of moving to standards-based design:

Validate, validate, validate

In order to make sure your page displays the way your DTD says it should, you will have to validate it. Some web development applications (like Dreamweaver) have validators built in, but there are a couple of on-line validators you can use as well.

W3C HTML Validation Service: http://validator.w3.org/
WDG Web Validator: http://www.htmlhelp.com/tools/validator/

Either of these tools will allow you to check a page on-line, and will also allow you to check local pages before you upload them to your site.

After your HTML validates, you need to make sure your CSS validates. If you are using a third-party tool like TopStyle, you can set the level of compliance to hit individual browsers or code to CSS1 or CSS2. Any errors will be color coded.

Once you have your style sheet completed, you can test it against the
W3C CSS Validation Service: http://jigsaw.w3.org/css-validator/

If your CSS validates, it's time for the last step. You need to check for 508 compliance with whatever accessibility tool is built into your development program, or use Cynthia Says or Bobby.

Cynthia Says: http://www.contentquality.com/
Bobby: http://bobby.watchfire.com

Both of these validators will identify any obvious accessibility problems, but will also give you some indications of what items must be checked manually.

There is another accessibility standard, the Web Access Initiative (WAI), which covers more issues but the issues are more difficult to test than the items in Section 508. While Missouri requires you to validate to Section 508, you may also want to test your pages against the WAI specifications. Both Cynthia Says and Bobby will test against either of the standards.

Browser Compatibility
In the days of the Browser Wars between Netscape and Internet Explorer, when new versions of browsers, each with its own unique feature set, would appear every few months, you had to be concerned with what browser to write to. Should you write for IE, since most of your users used it, and risk alienating your Netscape users (and your Mac users)? Or should you stick with HTML 3.2, since all the browsers knew what to do with pages written to that standard?

Well, the war may be over (IE won, for now), Microsoft hasn't updated their browser in a couple of years, the standards have stayed intact for several years, and other browsers, most notably those based on the Mozilla project, have worked very hard to implement those standards.

The result is that you don't have to talk about browser compatibility at all. All you need to be concerned about is accessibility. If you design your pages using standards and validate the pages before publishing, you will find that your pages will at least be readable on nearly any browser.